

12-30-94
FISH & RICHARDSON P.C.

4225 Executive Square
Suite 1400
La Jolla, California
92037

Telephone
858 678-5070

Facsimile
858 678-5099

Web Site
www.fr.com

December 28, 1999

Attorney Docket No.: 10559/128001/P7867

Box Patent Application

Assistant Commissioner for Patents
Washington, DC 20231

Presented for filing is a new original patent application of:

Applicant: GILBERT WOLRICH, DEBRA BERNSTEIN AND MATTHEW J.
ADILETTA

Title: PROVIDING REAL-TIME CONTROL DATA FOR A NETWORK
PROCESSOR

Enclosed are the following papers, including those required to receive a filing date
under 37 CFR 1.53(b):

	Pages
Specification	19
Claims	7
Abstract	1
Declaration	[To be Filed at a Later Date]
Drawing(s)	9

Enclosures:
— Postcard.

Basic filing fee	\$0
Total claims in excess of 20 times \$18	\$0
Independent claims in excess of 3 times \$78	\$0
Fee for multiple dependent claims	\$0
Total filing fee:	\$0

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EM040827355US

I hereby certify that this correspondence is being deposited with the
United States Postal Service as Express Mail Post Office to Addressee
with sufficient postage on the date indicated below and is addressed to
the Assistant Commissioner for Patents, Washington, D.C. 20231.

Date of Deposit December 28, 1999

Signature

Deborah L. Dean
Typed or Printed Name of Person Signing Certificate

12/28/99
JC583 U.S. PTO
Frederick P. Fish
1855-1930
W.K. Richardson
1859-1951

BOSTON
DELAWARE
NEW YORK
SILICON VALLEY
SOUTHERN CALIFORNIA
TWIN CITIES
WASHINGTON, DC

JC584 U.S. PTO
09/473571
12/28/99

FISH & RICHARDSON P.C.

Assistant Commissioner for Patents
December 28, 1999
Page 2

No filing fee is being paid at this time. Please apply any other required fees, **EXCEPT FOR THE FILING FEE**, to deposit account 06-1050, referencing the attorney document number shown above. A duplicate copy of this transmittal letter is attached.

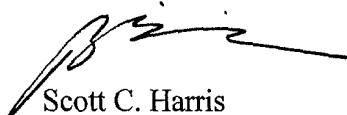
If this application is found to be incomplete, or if a telephone conference would otherwise be helpful, please call the undersigned at (858) 678-5070.

Kindly acknowledge receipt of this application by returning the enclosed postcard.

Please send all correspondence to:

SCOTT C. HARRIS
Fish & Richardson P.C.
4225 Executive Square, Suite 1400
La Jolla, CA 92037

Respectfully submitted,


Scott C. Harris
Reg. No. 32,030
Enclosures
SCH/rpi
10012704.doc

Bing ai
Reg. No. 43,312

APPLICATION
FOR
UNITED STATES LETTERS PATENT

TITLE: PROVIDING REAL-TIME CONTROL DATA FOR A
NETWORK PROCESSOR

APPLICANT: GILBERT WOLRICH, DEBRA BERNSTEIN AND
MATTHEW J. ADILETTA

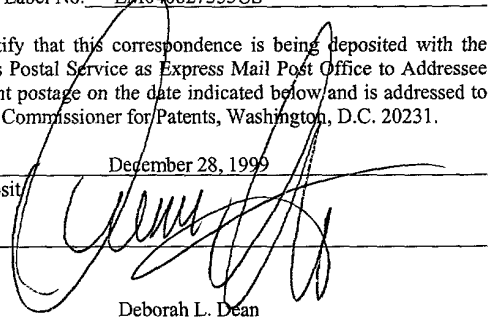
CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EM040827355US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Date of Deposit December 28, 1999

Signature


Deborah L. Dean

Typed or Printed Name of Person Signing Certificate

PROVIDING REAL-TIME CONTROL DATA
FOR A NETWORK PROCESSOR

Background of the Invention

5 This invention relates to controlling parallel processor arrays.

Many modern routers use application specific integrated circuits (ASIC's) to perform routing functions. The ASIC's can be designed to handle the protocols used by the networks
10 connected to the router. In particular, the ASIC's can route high provide high performance routing for data packets having one of a preselected set of protocols.

Summary of the Invention

15 According to one aspect of the invention, a processor includes a module configured to collect status data, one or more processing engines, and a push engine. The status data is collected from devices connected to a bus. The status data indicates readiness of the devices to participate in
20 data transfers over the bus. The processing engines schedule transfers of data packets between the processor and the devices. The push engine performs unsolicited transfers of a portion of the status data to the processing engines in response to the module collecting new status data.

25

Brief Description of the Drawings

FIG. 1 is a block diagram of a router based on a parallel processor;

FIG. 2 is a block diagram of a FIFO bus interface of
5 the parallel processor of FIG. 1;

FIG. 3 is a block diagram of one of the parallel processing engines used by the processor of FIG. 1;

FIG. 4 is a block diagram of a MAC port coupled to the parallel processor of FIG. 1;

10 FIG. 5A shows the status registers for receive-status data;

FIG. 5B shows the status registers for transmit-status data;

FIG. 5C shows the transmit FIFO buffer located in the
15 FIFO bus interface of FIG. 2;

FIG. 6 is a flow chart showing a process for providing ready-status data to scheduler threads;

FIG. 7 is a flow chart showing a process for collecting ready-status data from the MAC devices;

20 FIG. 8 is a flow chart for a process for transferring newly collected ready status data to the scheduler threads; and

FIG. 9 is a flow chart for a process that performs data transfers responsive to ready status data.

Description

FIG. 1 is a block diagram of a router 10 that uses a parallel processor 12, a set of media access chip (MAC) devices 14, 14', 14", and a FIFO bus 16. The router 10 performs data switching between source and destination networks 18, 18', 18" connected to the MAC devices 14, 14', 14". The MAC devices 14, 14', 14" are bridges that couple external networks 18, 18', 18" to the FIFO bus 16. The processor 12 can execute software to control data routing. By basing control on software, the processor 12 may be more easily modified to accommodate new protocols or data characteristics.

The router 10 performs data routing in two stages. First, one of the MAC devices 14, 14', 14" connected to the source network 18, 18', 18" transmits a data packet to the parallel processor 12 via the FIFO bus 16. Second, the parallel processor 12 retransmits the data packet over the FIFO bus 18 to the MAC device 14, 14', 14" connected to the destination network 18, 18', 18". The data transmissions over the FIFO bus 16 employ 64-byte data packets and proceed via an Ethernet protocol.

The parallel processor 12 has a parallel data forwarding structure that includes an array of identical processing engines 22a-22f. Each processing engine 22a-22f has an internal structure for executing a plurality of,

e.g., four, independent threads.

Referring to FIGs. 1 and 2, the processing engines 22a-22f process data packets received from the MAC devices 14, 14', 14". To process a data packet, one of the processing engines 22a-22f looks up routing information in a synchronous random-access memory (SRAM) 24 using information from the packet header. The processing engines 22a-22f also move the data packets from a FIFO buffer 58 to a queue in a synchronous dynamic random-access memory (SDRAM) 26. The FIFO buffer 58 temporarily stores data packets received from the MAC devices 14, 14', 14". The various queues located in the SDRAM 26 are classified by destination MAC device 14, 14', 14" and retransmission priority.

The processing engines 22a-22f also process data from the queues of the SDRAM 26. This processing includes moving data packets from the queues of the SDRAM 26 to a FIFO buffer 60. The FIFO buffer 60 temporarily stores data prior to retransmission to the MAC devices 14, 14', 14" over the FIFO bus 16. Along with the data, associated control and destination information are stored in the FIFO buffer 60 for use in transmitting the data. The associated data is 16 bytes wide.

The SRAM 24 and SDRAM 26 couple to the processing engines 22a-22f through respective SRAM and SDRAM controllers 34, 36. The SRAM controller 34 has content

addressable memory that supports look ups of identification information on the queues of the SDRAM 24. The look-ups use header data from received data packets. The SDRAM controller 36 coordinates data writes to and reads from the queues of the SDRAM 24 that store received data packets.

The parallel processor 12 has several internal busses 39, 40, 41. An S bus 39 couples the processing engines 22a-22f to a FIFO bus interface 38 (FBI) and to the SRAM controller 34. An M bus 40 couples the processing engines 22a-22f and the FBI 38 to the SDRAM controller 36 and the SDRAM 26. An AMBA bus 41 couples a processor core 44 to the processing engines 22a-22f and the FBI 38.

The FBI 38 controls data transfers on the FIFO bus 16 and collects status data on the readiness of the ports 28, 30, 32 of the MAC devices 14, 14', 14" to participate in data transfers over the FIFO bus 16. The ready status data is collected from the MAC devices 14, 14', 14" through a ready bus 42, which is also controlled by the FBI 38.

Referring again to FIG. 1, the processor core 44 uses software to perform a variety of functions. The functions may include data packet routing, exception handling, queue management, monitoring of data packet transfers, supporting network management protocols and/or providing local area network emulation.

The parallel processor 12 includes a PCI bus interface

46 that couples to a PCI bus 48. The PCI bus 48 can support communications between the parallel processor 12 and external processors. The other processors may control and/or reprogram the processor core 44 or other components 22a-22f, 38 of the multiprocessor 12.

Referring again to FIG. 2, the connections between the FBI 38 and the processing engines 22a-22f are shown. The FBI 38 includes a control module 50 for the ready bus 42 and a push engine 62. The control module 50 periodically collects receive-ready status data and transmit-ready status data from the MAC devices 14, 14', 14". The collected ready status data is stored in a set of status registers 54. The set includes separate registers for storing receive-ready status data and transmit-ready status data. The push engine 62 regularly sends the ready status data over the S bus 39 to scheduler threads located in the processing engines 22a-22f in response to commands from logic internal to the FBI 38.

The processing engines 22a-22f include separate receive-scheduler and transmit-scheduler threads. The receive-scheduler thread schedules the processing of data received from the FIFO bus 16. The transmit-scheduler thread schedules the processing of data to be transmitted to the FIFO bus 16.

The receive-scheduler thread assigns data forwarding

and header processing tasks to other threads in the processing engines 22a-22f. These tasks include sharing operation of a push engine 62 that transports data from the receive FIFO buffer 58 in the FBI 38 to one of the storage queues in the SDRAM 26.

The transmit-scheduler thread also assigns data forwarding tasks to other threads in the processing engines 22a-22f. These tasks include sharing in operation of a pull engine 64, which moves data from the storage queues in the SDRAM 26 to the transmit FIFO buffer 60. The tasks also include directing the pull engine 62 to write transmission control and MAC device 14, 14', 14" address information to the FIFO buffer 60. Each data packet in the transmit FIFO buffer 60 has associated address and control information that control the retransmission over the FIFO bus 16.

To control data forwarding by the push and pull engines 62, 64, the execution threads of the processing engines 22a-22f send commands signals to FIFO command queues 66, 68 via a line 70. Components of the FBI 38 can also send commands to the command queues 66, 68 of push and pull engines 62, 64. For example, the ready bus controller 50 can send a command to the queue 66 that causes the push engine 62 to transfer ready status data from the status registers 54 to the processing engines 22a-22f. An arbiter 56 controls transmission of commands from the queues 66, 68 to the push

and pull engines 62, 64.

The push and pull engines 62, 64 perform several types of tasks. The push and the pull engines 62, 64 are involved in bi-directional forwarding of data packets between the
5 FIFO buffers 58, 60 and the SDRAM controller 36. The push and pull engines 62, 64 also operate a large hardware unit 71 located in the FBI 38. The push engine 62 also forwards ready status data from the set of status registers 54 to the receive- and transmit-scheduler threads located in the
10 processing engines 22a-22f.

The hardware unit 71 performs various operations for the execution threads of the processing engines 22a-22f and includes a hash unit 72 and a scratchpad memory 73. The execution threads operate the hardware unit 71 by sending
15 commands to the queues 66, 68. To perform the operations, the pull engine 64 retrieves input data over the S bus 39 from output transfer registers 80a-80f of the requesting processing engine 22a-22f. The pull engine 64 moves the retrieved data and associated commands to the hardware unit
20 71. The hardware unit 71 forwards results from the operations to the push engine 62. The push engine 62 uses command information from the command queue 66 and/or pull engine 64 to transmit the results back over the S bus 39 to input transfer registers 78a-78f of the requesting or
25 destination processing engine 22a-22f.

Referring to FIG. 3, one embodiment 74 of the processing engines 22a-22f is shown. The processing engines 22a-22f have input/output terminals 75-77 for control signals, address signals, and data. Control signals, address signals, and data are transferred to and from the processing engines 22a-22f over three busses, i.e., the M bus 40, the S bus 39, and the AMBA bus 41. The address signals identify both a processing engine 22a-22f and an execution thread so that external commands can independently address different threads. Data is received at and transmitted from respective input and output transfer registers 78, 80. Each input and output transfer register 78, 80 is assigned to an individual execution thread. To write data to or read data from a particular execution thread, an external device accesses one of the transfer registers 78, 80 assigned to the particular thread.

Referring to FIG. 4, the port 28 of the MAC device 14 is shown. The port 28 has transmit and receive FIFO buffers 90, 92 for storing data prior to transmission to and after reception from the FIFO bus 16, respectively. Both buffers 90, 92 have entries of fixed size that are multiples of 64 bytes, i.e., the size of data packets on the FIFO bus 16. The port 28 also includes address decoders and a controller 94. The controller 94 controls both protocol transfers over the FIFO bus 16 and responses to ready status queries from

the ready bus 42. The responses to the queries indicate whether the transmit buffer 90 has a 64 byte data packet to transmit and/or whether the receive buffer 92 has space to receive a 64 byte data packet.

5 The various ports 28, 30, 32 of the MAC devices 14, 14', 14" may support different data transfer rates. The ports 28, 30 of the MAC devices 14, 14' support transfer rates of about ten or one hundred megabits of data per second. The port 32 of the MAC device 14" may have a
10 transfer rate of up to about one gigabit per second.

 The ready bus 42 includes control/address and data lines. The control/address lines enable selection of a transaction type and a port 28, 30, 32 of the MAC devices 14, 14', 14". The data line transfers receive-ready status
15 data and transmit-ready status data to the FBI 38 in response to status queries from the control module 50 for the ready bus 42.

 Referring to 5A, the registers R_1 , R_2 , R_3 that store receive-ready status data are shown. The registers R_1 and
20 R_2 store receive-ready status data for individual MAC ports 28, 30, 32. The readiness of each MAC port 28, 30, 32 to transmit a data packet to the FIFO bus 16 is indicated by the value of an associated bit or flag stored in one of the registers R_1 , R_2 . One logic value of the bit or flag
25 indicates that the associated port 28, 30, 32 has a data

packet ready to transmit, and the other logic value indicates the associated port 28, 30, 32 has no ready data packets. Different ports 28, 30, 32 may have data packets of different sizes, but the receive scheduler thread knows
 5 the packet size associated with each port 28, 30, 32.

The registers R_2 and R_3 have 32 bits each and thus, can accommodate receive-ready status data for up to 64 different MAC ports 28, 30, 32.

The register R_3 stores a cyclic counter value, which
 10 acts as a time stamp for the receive-status data stored in registers R_1 , R_2 . The counter value is incremented each time new receive-status data is collected. By comparing the counter value to a previously received counter value, the scheduler thread can determine whether the present receive-
 15 status data is new or stale, i.e., whether the data has already been seen.

Referring to FIG. 5B, the registers R_4 , R_5 , R_6 that store transmit-ready status data are shown. The registers R_4 and R_4 store transmit-ready status data for individual
 20 MAC ports 28, 30, 32. Each MAC port 28, 30, 32 has an associated bit or flag in one of the registers R_4 and R_4 . One logic value of the bit or flag indicates that the associated port 28, 30, 32 has enough space to receive a data packet, and the other logic value indicates the
 25 associated port 28, 30, 32 does not have enough space.

The registers R_4 and R_5 have a total of 64 bits and thus, can report transmit ready status for up to 64 MAC ports 28, 30, 32.

Referring to FIG. 5C, the number stored in register R_6 indicates the position of a remove pointer 96 in the transmit FIFO buffer 60. For an embodiment in which the transmit FIFO buffer 60 has sixteen entries, the position of the remove pointer is represented as a 4-bit number.

Since the FBI 38 transmits 64-byte data packets from the buffer 60 according to a FIFO scheme, the remove pointer 96 indicates which data packets are scheduled but not transmitted. The position of the pointer 96 can be used to determine which MAC ports 28, 30, 32 have been scheduled to receive a data packet but have not yet received a data packet. Such ports 28, 30, 32 may have status data in registers R_4 , R_5 indicating an availability to receive a data packet even though the available space has already been assigned to a waiting data packet.

The transmit scheduler thread can use the position of the remove pointer 96 to interpret transmit-ready status data of the registers R_4 , R_5 . From the position of the remove pointer 96, the transmit scheduler thread identifies MAC ports 28, 30, 32 already scheduled to receive a data packet. The transmit scheduler thread does not schedule a new data packet for such ports, because the waiting and

already scheduled data packet may take the available space therein.

In the parallel processor 12, the collection of ready status data is asynchronous with respect to scheduling of data packet transfers. The asynchronous relationship enables both the collection of ready status data and the scheduling of data packets to have higher effective bandwidths. The asynchronous relationship also introduces some unpredictability into latencies associated with the transfer of newly collected ready status data to scheduler threads.

Referring to FIG. 6, a process 100 by which the FBI 38 provides ready status data to the scheduler threads is shown. The FBI 38 performs 102 a collection cycle in which new ready status data is obtained from the MAC devices 14, 14', 14" interactively via the ready bus 42. In response to completing the collection cycle, the FBI 38 performs an unsolicited transfer 104 of the newly collected ready status data to the input transfer registers 78a-78f assigned to the scheduler threads. In an unsolicited data transfer, the destination device for the transfer does not request the transfer. The transfer of ready status data from the FBI 38 to destination processing engines 22a-22f and scheduling threads proceeds without any request from the processing engines 22a-22f. Instead, the FBI 38 automatically performs

the transfer in response to finishing a collection cycle for the ready status data. The completion of each collection cycle causes issuance of a command to the push engine 62, which transfers the ready bus data to the processing engines 22a-22f. After completing the transfer, the FBI 38 loops back 106 to collect new ready status data.

Making transfers of new ready status data unsolicited lowers latencies for delivering such data to scheduler threads. Since latencies in delivering such data can cause scheduling errors, making the transfer of ready status data unsolicited can lower numbers of occurrences of scheduling errors.

Referring to FIG. 7, a process 110 by which the FBI 38 collects ready status data is shown. Separate collection cycles are performed to collect receive-ready status data and to collect transmit-ready status data. Each collection cycle also initiates an unsolicited transfer of at least a portion of the collected ready status data to the processing engines 22a-22f.

To start a new collection cycle, the control module 50 for the ready bus 42 selects 112 the addresses to be polled for ready status data. The selection may be for all addresses of the MAC ports 28, 30, 32 connected to the FIFO bus 16 or for a sub-range of the addresses. If a sub-range is selected, the collection of new ready status data spans

several cycles, a portion of the MAC ports 28, 30, 32 being polled in each cycle. The sub-range polled in collection cycles may be programmed into the processor core 44 or the FBI 38.

5 The control module 50 polls 114 by sending status queries over the ready bus 42 to the selected ports 28, 30, 32 of the MAC devices 14, 14', 14''. In response to the queries, the control module 50 receives 116 new ready status data from the polled ports 28, 30, 32. A response to a
10 query for receive-ready status data indicates whether the responding port 28, 30, 32 has a data packet ready to transmit. A response to a query for transmit-ready status indicates whether the responding port 28, 30, 32 has space available to receive another data packet.

15 The control module 50 writes 118 new ready status data, which has been from the responses, to the status registers R_1 , R_2 , R_4 , R_5 , shown in FIGs. 5A-5B. The control module 50 also increments 120 the counter value in status register R_3 .

Incrementing the counter value updates the time stamp
20 associated with the newly collected ready status data. After updating the time stamp, the FBI 38 performs an unsolicited transfer of the newly collected ready status data to the scheduler threads located in processing engines 22a-22f.

25 The FBI 38 transmits 126 data packets from the transmit

FIFO buffer 60 asynchronously with respect to the collection of ready status data from the MAC devices 14, 14', 14''. In response to each transmission, the FBI 38 advances 128 the remove pointer 96 of the transmit FIFO buffer 60 and writes
 5 130 the new position of the remove pointer 96 to status register R_6 . The number stored in the status register R_6 reflects the present position of the remove pointer 96 of the transmit FIFO buffer 60.

Referring to FIG. 8, a process 140 by which the FBI 38
 10 transfers receive-ready and transmit-ready status data to the respective receive and transmit scheduler threads is shown. The FBI 36 transfers the ready status data via the S bus 39 when the S bus 39 is not being used for communications with the SRAM controller 34.

15 Completion of a collection cycle enables 142 the push engine 62 to transfer ready status data from the status registers 54 to the appropriate execution threads, i.e., scheduler threads. The push engine 62 reads 144 both a value for the number of the status registers R_1 - R_3 or R_4 - R_6
 20 to be transferred and the identity of the target scheduler thread. One, two, or three status registers may be transferred in one cycle. The count and identity of the scheduler threads, i.e., for both the thread and the associated processing engine 22a-22f, are stored in control
 25 registers 52.

Transfers of 1, 2 or 3 of the status registers R_1 - R_6 write to the 1, 2, or 3 lowest consecutive input transfer registers 78a-78f assigned to the target scheduler thread. But, the transfers may also alternate targeting of the input transfer registers 78a-78f. To alternate targets, the push engine 62 sends consecutive transfers to different input transfer registers 78a-78f assigned to the same scheduler thread. For example, a first transfer of two of the status registers R_1 - R_3 could be written to the two lowest input transfer registers, and the next transfer would then be written to the two next-lowest input transfer registers. From the count and alternate-select status, the push engine 62 determines which input transfer registers 78a-78f to write during the transfer.

The push engine 62 transmits a transfer protect control signal to the target input transfer registers 78a-78f. The transfer protect signal protects the target transfer registers 78a-78f against read-write conflicts during transfers. The transfer protect signal blocks reads of the registers 78a-78f by the associated scheduler threads. While protected from such reads, the push engine 62 writes the new ready status data to the input transfer registers 78a-78f.

After completing a transfer of ready status data, the push engine 62 stops transmitting the transfer protect

signal. When the protect signal is no longer asserted, the scheduler threads can read the input transfer registers 78a-78f. The scheduler threads reads the ready status data from the input transfer registers 78a-78f in the order written to avoid other read-write conflicts.

Referring to FIG. 9, a process 160 that performs data transfers responsive to ready status data is shown. To schedule a transfer, the appropriate scheduler thread determines 162 which MAC ports 28, 30, 32 are available from the values of the new ready status data. For the available ports 28, 30, 32, the scheduling thread selects 164 an available execution thread to handle the data transfer and signals the selected thread. The selected thread and FBI 38 perform the scheduled data transfer 166 the scheduled data transfer.

For receive-ready status data, the scheduler thread also compares the time stamp of the status data to the time-stamp of time stamps of previous receive-ready status data.

If the time stamp has an old value the ready status data is stale, and the receive scheduler thread stops without scheduling data transfers. Otherwise, the receive scheduler thread proceeds as described above.

For transmit-ready status data, the scheduler thread uses present values of the remove pointer 96 to determine whether any of the available ports are already scheduled to

receive a data packet. Any such devices are not scheduled for another data transmission.

While various embodiments have been described in the detailed description, the description is intended to

5 illustrate and not to limit the scope of the invention,
which is defined by the appended claims. Other aspects,
advantages, and modifications are within the scope of the
claims.

What is claimed is:

10

1. A processor, comprising:

a module configured to collect status data from devices connected to a bus, the status data indicating readiness of the devices to participate in data transfers over the bus;

5 one or more processing engines to schedule transfers of data packets between the processor and the devices; and

a push engine to perform unsolicited transfers of the status data to the processing engines in response to the module collecting new status data.

10 2. The processor of claim 1, wherein the processing engine comprises:

one or more input transfer registers to receive the unsolicited transfers of status data for use to schedule the
15 transfers of data packets.

3. The processor of claim 2, wherein the processing engine uses a portion of received new status data to schedule retrievals of data packets from the devices.

20 4. The processor of claim 2, wherein the processing engine uses a portion of the received status data to schedule transmissions of data packets.

25 5. The processor of claim 4, wherein the processing

engine uses a portion of the received status data to determine whether schedule transmissions of data packets have been completed.

5 6. The processor of claim 1, wherein the module is configured to poll the devices for the status data over a second bus.

10 7. The processor of claim 2, wherein a portion of the status data are flags indicative of whether associated devices have data packets to transmit.

15 8. The processor of claim 2, wherein a portion of the status data includes flags indicative of whether associated devices have space to receive data packets.

 9. A method of transferring data packets over a bus, comprising:

 collecting information on readiness of devices
20 connected to the bus to one of transmit and receive data packets; and

 transferring a portion of the collected information to a processing engine configured to schedule data transfers, the transferring being unsolicited by the processing engine.

25

10. The method of claim 9, further comprising:
scheduling data transfers with a portion of the devices
based on the transferred portion of the collected
information.

5

11. The method of claim 10, wherein scheduling further
includes:

determining whether the transferred information is at
least partly new; and

10 wherein the scheduling is performed in response to the
transferred information being at least partly new.

12. The method of claim 10, wherein determining
includes comparing a value of a time stamp transferred with
15 the information to a previous value of the time stamp.

13. The method of claim 10, wherein scheduling further
comprises:

determining whether an earlier scheduled data transfer
20 have been completed from the transferred information.

14. The method of claim 10, wherein collecting further
comprises:

polling the devices for ready status data on the
25 availability of ports thereon; and

receiving ready status data associated with individual ones of the devices in response to the polling.

15. The method of claim 12, wherein collecting further
5 comprises:

writing the received ready status data to a status register

scheduling transfers of data packets over the bus in response to the transferred portion of the ready status
10 data.

16. The method of claim 9, wherein the transferred portion of the information includes flags that indicate whether associated ports of the devices have one of space to
15 receive data packets and data packets ready to transmit over the bus.

17. The method of claim 16, further comprising:
polling the ports of the devices over a second bus to
20 determine values of the flags.

18. A router, comprising:
a bus; and
a parallel processor coupled to the bus and comprising:
25 a plurality of processing engines to process data

transfers with a plurality of devices connected to the bus;
and

an interface connected to collect ready status data
from the devices and to automatically transfer ready status
5 data the processing engines in response to the status data
being collected.

19. The router of claim 18, wherein the ready status
data indicates the readiness of individual ones of the
10 devices to one of receive a data packet from and transmit a
data packet to the parallel processor.

20. The router of claim 18, wherein the ready status
data includes a time stamp indicative of a staleness of the
15 ready status data.

21. The router of claim 18, wherein a portion of the
ready status data includes information to enable the
processing engines to identify which scheduled data
20 transfers to the devices have been completed.

22. The router of claim 18, further comprising:
a ready bus capable of transferring ready status data
from the devices to the interface.

25

23. The router of claim 19, wherein the ready status data indicates whether associated ports of the devices are ready to perform one of a transmission of a data packet to the bus and a receive of a data packet from the bus.

5

24. The router of claim 20, wherein each processing engine comprises at least one input transfer register; and the interface is configured to write ready status data to one of the input transfer registers assigned to a scheduler thread.

10

25. The router of claim 24, wherein the interface is configured to protect one of the input transfer registers from being read by the processing engines during the transferring of ready status data thereto.

15

26. The router of claim 18, wherein the devices are capable of transmitting data packets between the bus and external networks.

20

27. The router of claim 18, wherein the interface transfers the collected status data without being solicited to transfer the data by the processing engines.

25

28. An article comprising a computer-readable medium

which stores executable instructions for transferring data packets over a bus, the instructions causing a processor to:

collect information on readiness of devices connected to the bus to one of transmit and receive data packets; and

5 transfer a portion of the collected information to a processing engine configured to schedule data transfers, the transferring being unsolicited by the processing engine.

29. The article of claim 28, the instructions further
10 causing the processor to:

schedule data transfers with a portion of the devices based on the transferred portion of the collected information.

30. The article of claim 29, the instructions further
15 causing the processor to:

determine whether the transferred information is at least partly new; and

wherein instructions causing the processor to schedule
20 are performed in response to determining that the transferred information being at least partly new.

PROVIDING REAL-TIME CONTROL DATA
FOR A NETWORK PROCESSOR

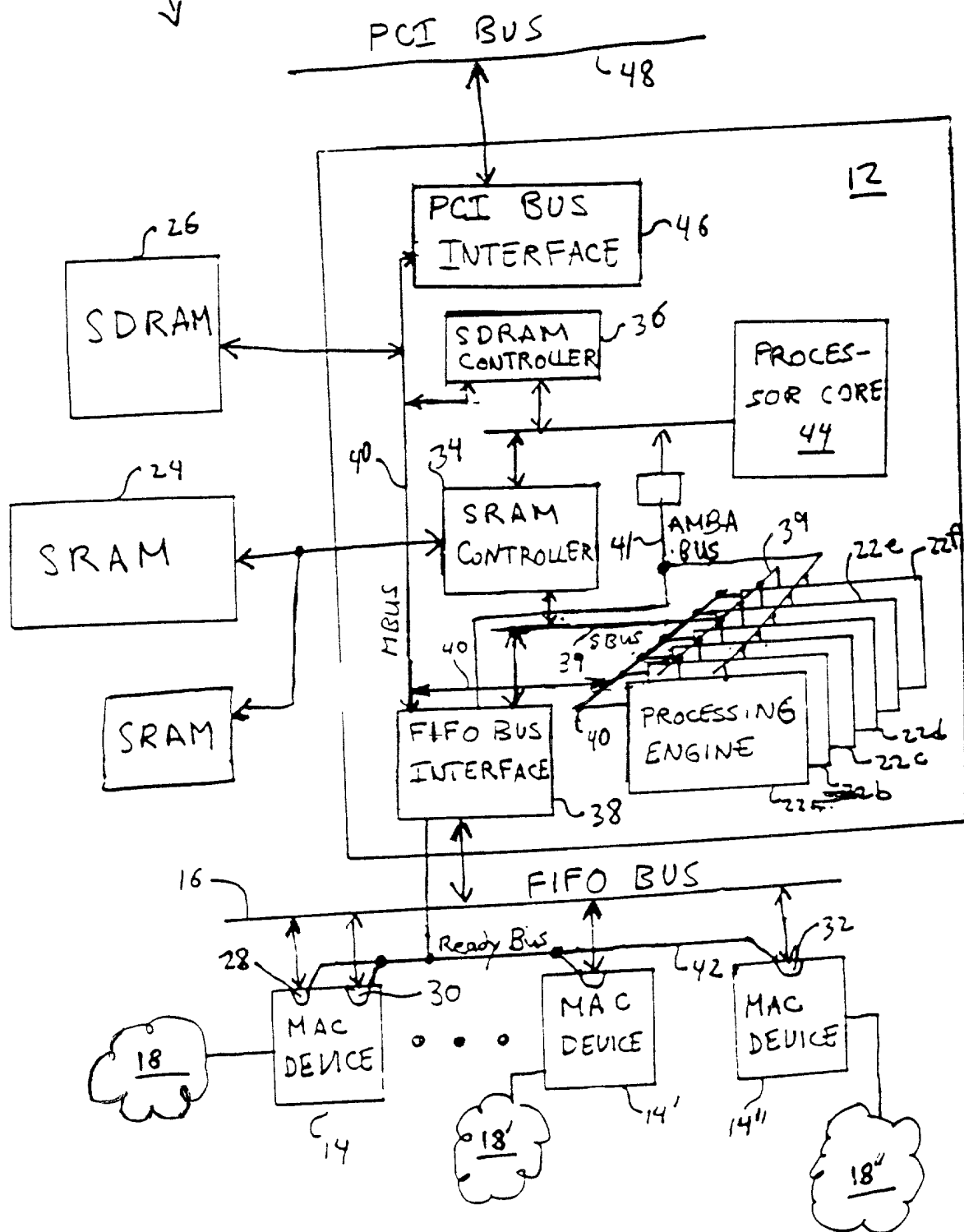
5

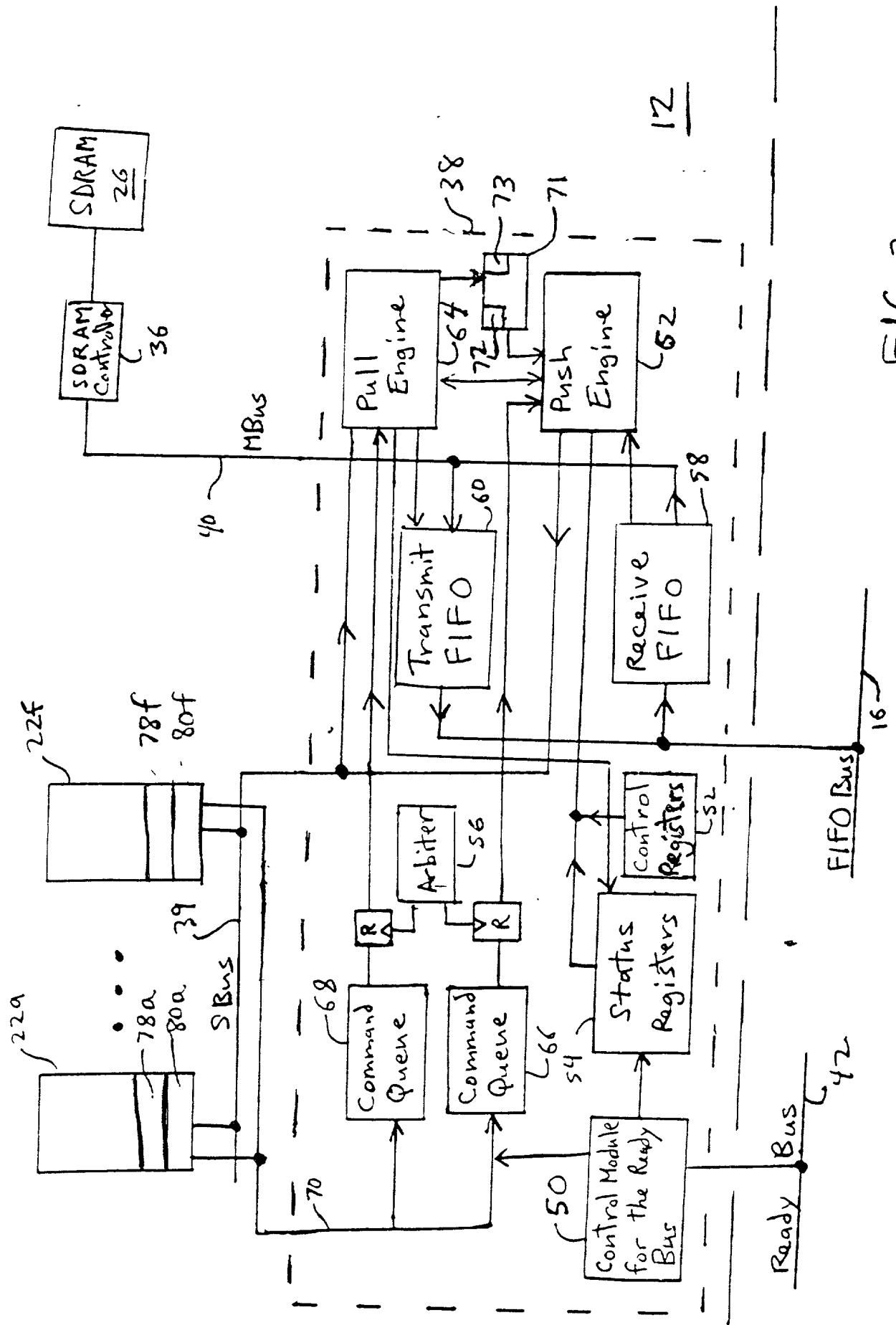
Abstract of the Disclosure

A multiprocessor includes a module configured to collect status data, one or more processing engines, and a push engine. Part of the status data is collected from devices connected to a bus and indicates readiness of the devices to participate in data transfers over the bus. The processing engines schedule transfers of data packets between the processor and the devices. The push engine performs unsolicited transfers of a portion of the status data to the processing engines in response to the module collecting new status data.

200007602

10
↘





F16.2

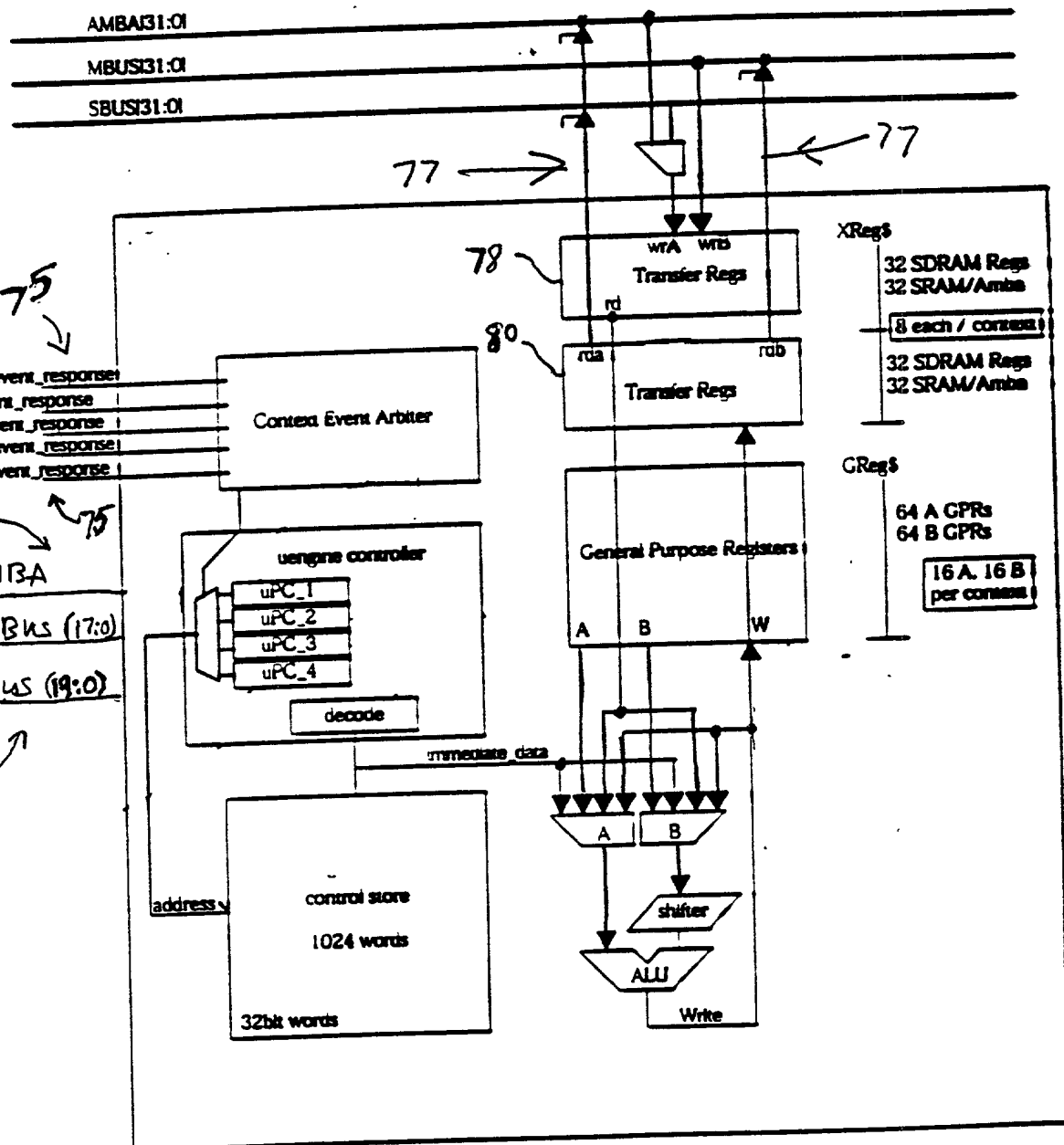


FIG. 3

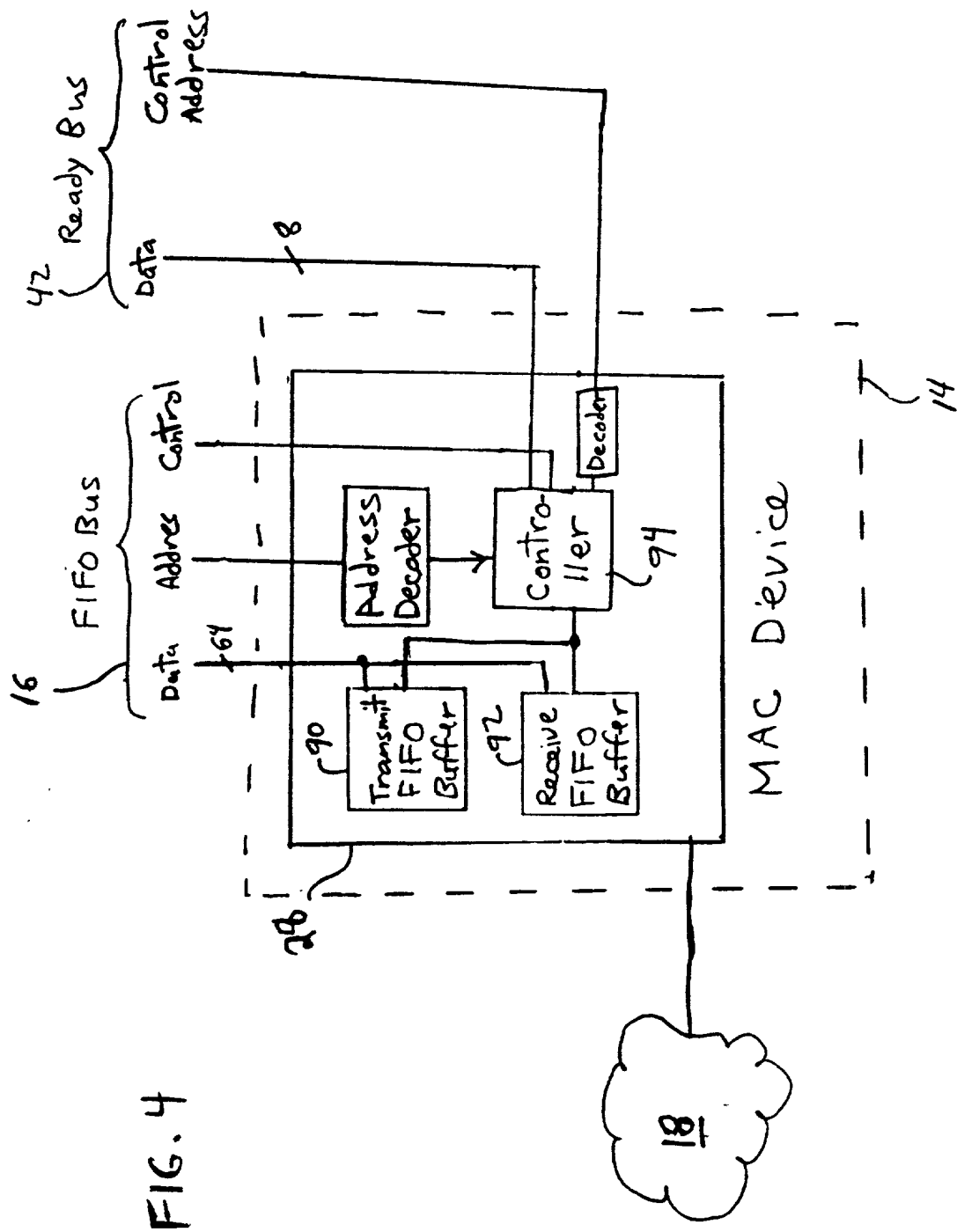


FIG. 4

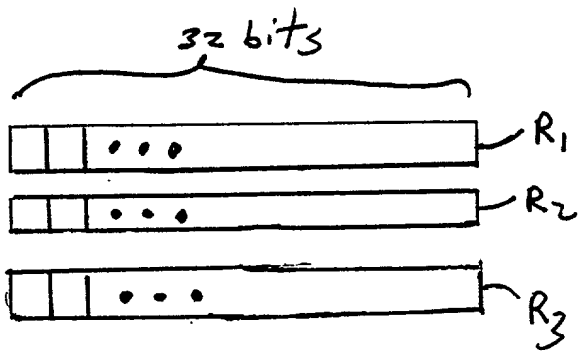


FIG. 5A

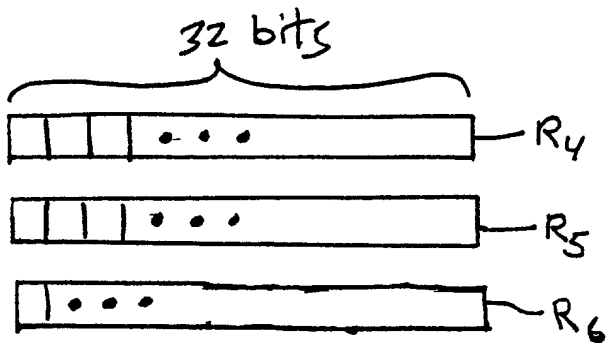


FIG. 5B

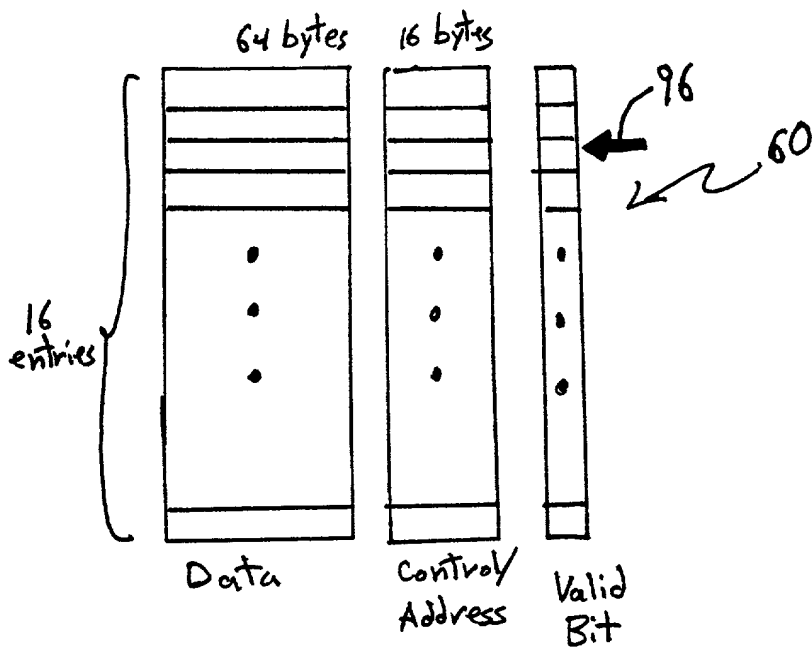


FIG. 5C

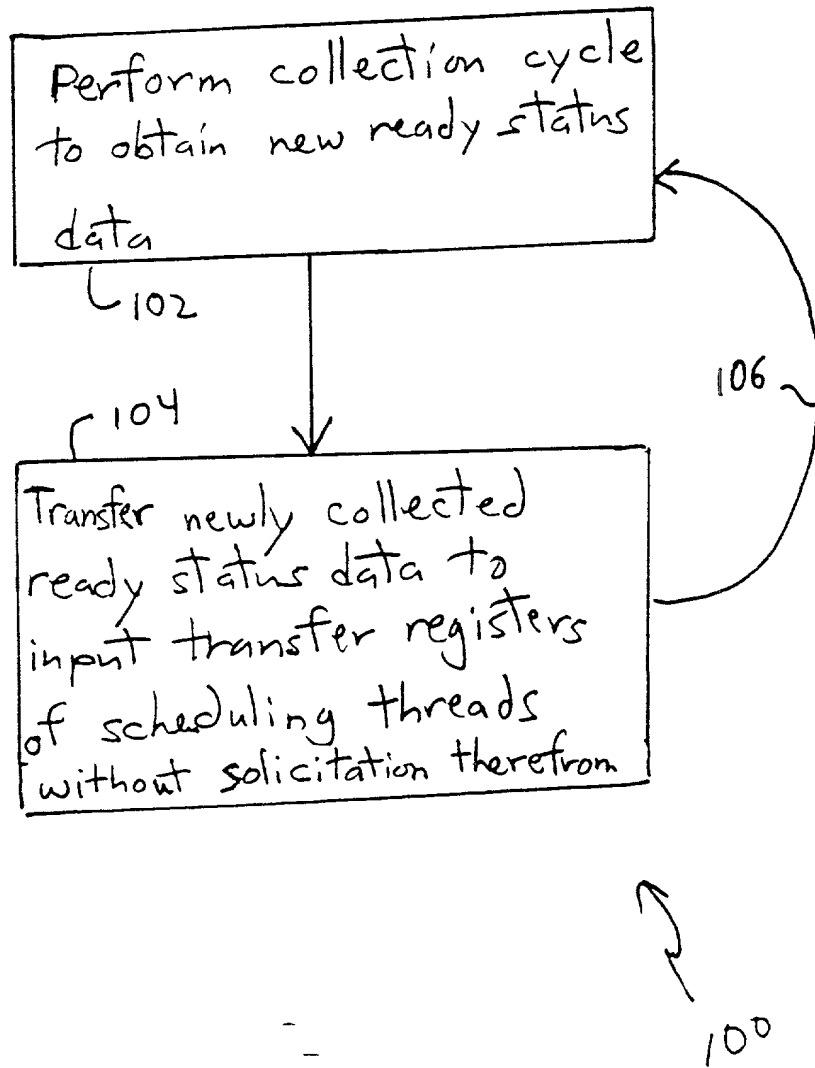
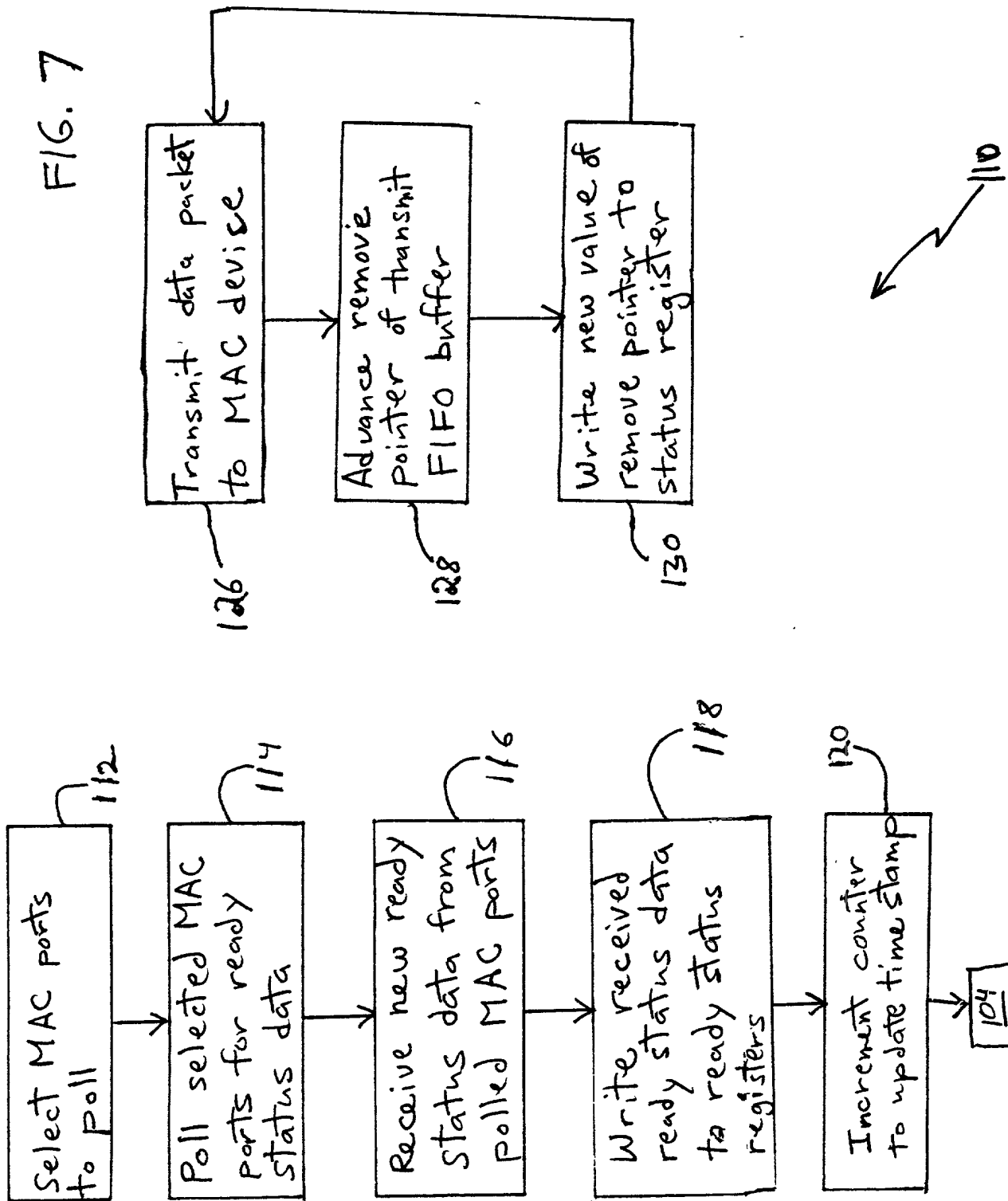


FIG. 6

FIG. 7



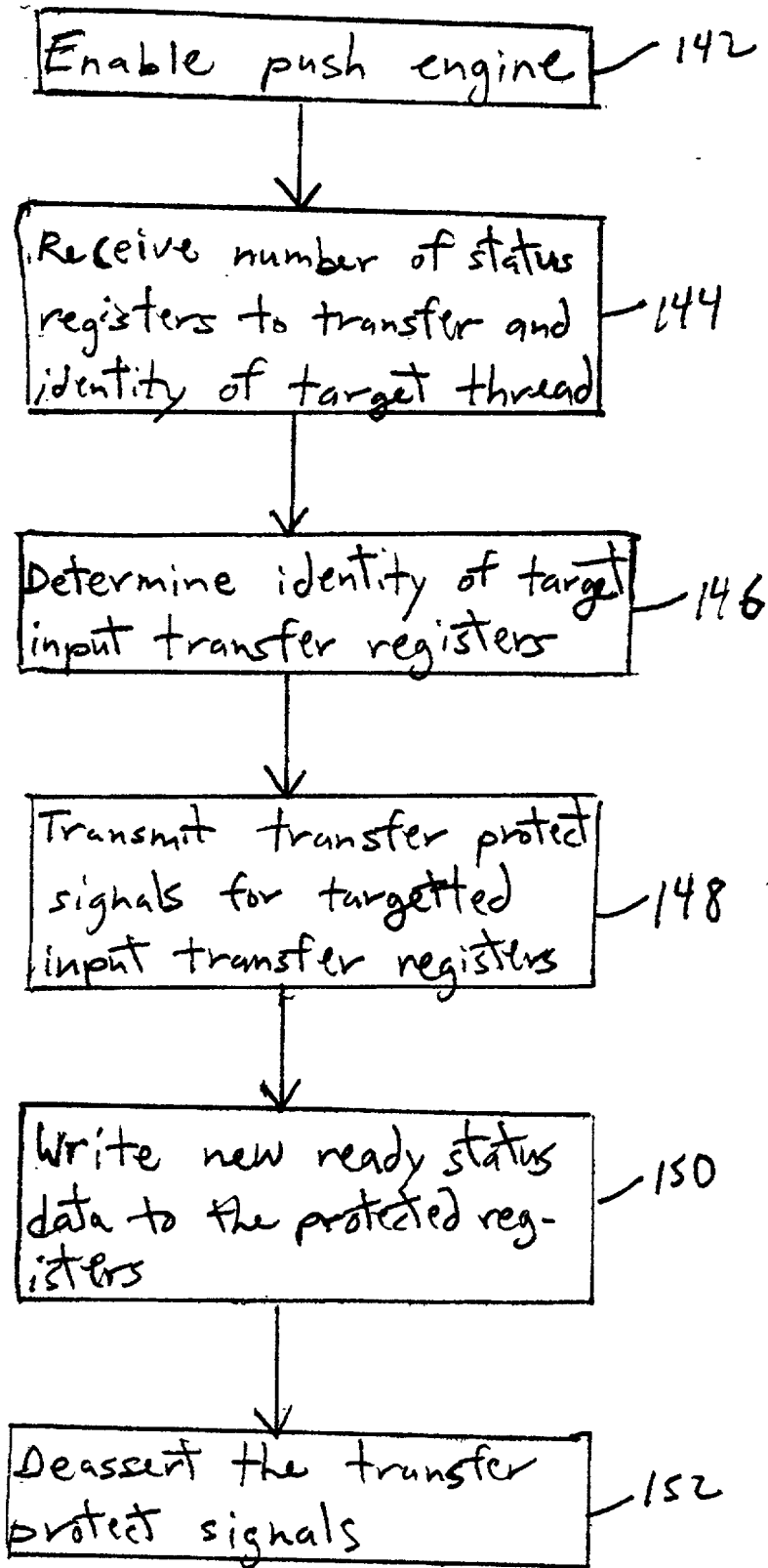
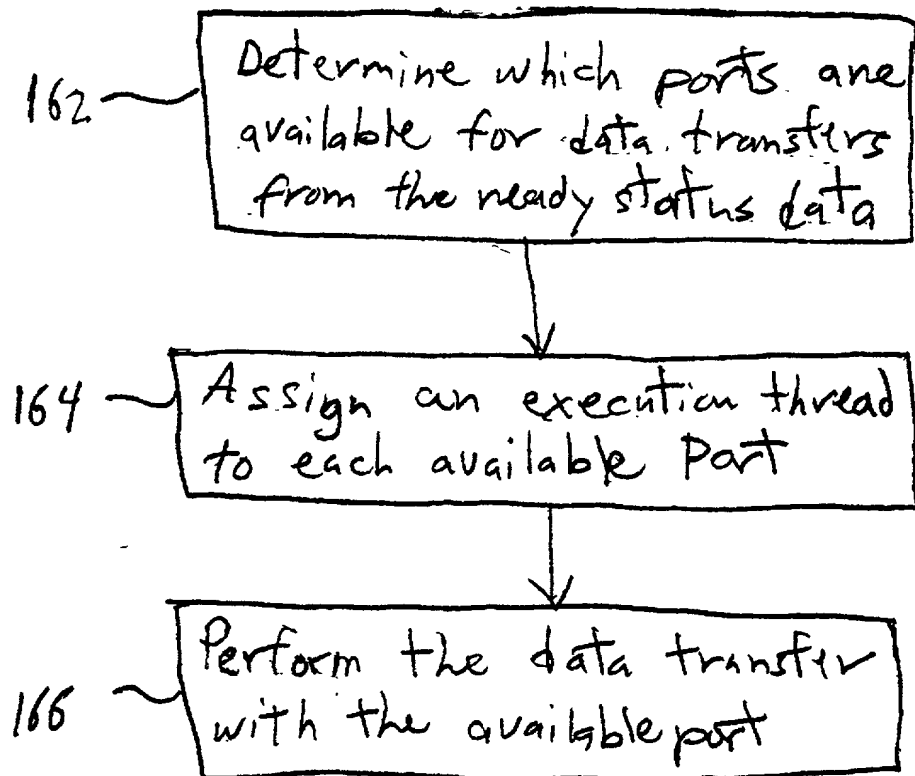


FIG. 8



160

FIG. 9